
Semantik

Semantik von Programmiersprachen

Prof. Tobias Nipkow, Ph. D.

Erstellt von Benjamin Gufler

Inhaltsverzeichnis

0.1	Ziele	1
0.2	Aufbau der Vorlesung	1
0.3	Puzzles	1
1	Einführung	5
1.1	Semantik von Ausdrücken	6
2	Operationale Semantik	7
2.1	Big – Step Semantik (natural evaluation)	7
2.2	Regelinduktion	8
2.3	Small – Step – Semantik (structural operations)	10
2.4	Erweiterungen von While	12
2.4.1	Nichtdeterminismus	12
2.4.2	Parallelismus mit gemeinsamen Variablen	13
2.4.3	Lokale Variablen	13
2.5	Ein Compiler	13
3	Induktive Definition und Regelinduktion	17
3.1	Regelinduktion	17
3.2	I_R als kleinster Fixpunkt	18

0.1 Ziele

- deskriptiv: Wie beschreibt man Semantik?
- analytisch: Wie benutzt man Semantik zur Analyse von
 - Programmiersprachen:
 - * deterministisch?
 - * typsicher?
 - Werkzeuge:
 - * Compiler-Korrektheit
 - Programme:
 - * Korrektheit
 - * Laufzeitanalyse

Beispielsprache: While

```

s ::= skip
    | x := a
    | s; s
    | if b then s else s
    | while b do s
  
```

+ Erweiterungen

0.2 Aufbau der Vorlesung

1. operationale Semantik

$\langle \text{Anweisung, Zustand} \rangle \rightarrow \text{Zustand}' \quad \langle s, \sigma \rangle \rightarrow \sigma'$

$$\frac{B[b]\sigma = \text{true} \quad \langle s, \sigma \rangle \rightarrow \sigma'' \quad \langle w, \sigma'' \rangle \rightarrow \sigma'}{\langle \text{while } b \text{ do } s(=w), \sigma \rangle \rightarrow \sigma'}$$

2. Definition einer Teilsprache (\gg Jinja \ll) von Java mit operationaler Semantik

- Quellsprache
- abstrakte Maschine (JVM)
- Compiler

3. axiomatische Semantik (Hoare – Logik)

$\{P\} S \{Q\}$

0.3 Puzzles

- Finde Körper, so dass 1 ausgegeben wird.

```

1  var x: integer;
2  x := 0;

4  function f(): integer
   begin
6      x := x + 1;
       f := x;
   
```

```

8  end
10 if f()= f() then
    print 0
12 else
    print 1

```

- Finde Ausdruck e , so dass $e + f()$ und $f() + e$ sich verschieden verhalten.

```

1  function f(): integer
2  begin
    f := f()
4  end

```

Lösung: $e = \frac{1}{0}$

- Gilt $\{ \} x := 0 \{x = 0\}$?
Ja.
- Gilt $\{ \} A[1] := 0 \{A[1] = 0\}$?
Ja.
- Gilt $\{ \} A[A[1]] := 0 \{A[A[1]] = 0\}$?
Nein: Wenn z.B. vorher $A[1] = 1$ gilt, dann nicht.
- Finde Kontext, der $x := 0$; $y := 1$ und $y := 1$; $x := 0$ unterscheidet.

```

1  procedure p( var x, y: integer )
2  begin
    // Anweisungsblock
4  end

6  var z: integer;
    p( z, z )

```

»Aliasing is hell!«

- Ist q äquivalent zu $\langle s \rangle$?

```

1  proc p( proc r )
2  begin
    end
4
    proc q()
6  var x: integer;
    proc r()
8      begin
        x := x + 2
10     end
    begin
12     x := 0;
        p( r );
14     if even( x ) then
        <s>
16     end

```

Ja, falls p terminiert oder $\langle s \rangle$ nicht terminiert. (Beweis?)

- Ist $x := x$ äquivalent zu `skip`?
 $x := x \parallel x := x+1$
Laufzeitfehler, falls x nicht initialisiert ist.
- Finde Kontext, der $1 + 2$ und $2 + 1$ unterscheidet.
Ausdruck * 5
- Ausdruck

Kapitel 1

Einführung

Mengen	Meta – Variablen
\mathbb{Z}	i, j, m, n
Var (Programmvariablen)	x, y, z
$T = \{\text{true}, \text{false}\}$	

Abstrakte Syntax Arithmetische Ausdrücke $Aexp$, $a \in Aexp$

$$\begin{aligned} a ::= & n \\ & | x \\ & | a_1 \text{ aop } a_2 \\ & \text{aop} \in \{+, -, *\} \end{aligned}$$

Boolesche Ausdrücke $Bexp$, $b \in Bexp$

$$\begin{aligned} b ::= & \text{true} \\ & | \text{false} \\ & | a_1 \text{ rop } a_2 \\ & | b_1 \text{ bop } b_2 \\ & | \neg b \\ & \text{rop} \in \{=, \leq\} \\ & \text{bop} \in \{\wedge, \vee, \dots\} \end{aligned}$$

Anweisungen Stm , $s \in Stm$

$$\begin{aligned} s ::= & \text{skip} \\ & | x := a \\ & | s_1; s_2 \\ & | \text{if } b \text{ then } s_1 \text{ else } s_2 \\ & | \text{while } b \text{ do } s \end{aligned}$$

Konkrete Syntax definiert Strings; sollte eindeutig sein: höchstens ein Ableitungsbaum pro String. Die oberhalb definierte Syntax ist *nicht* eindeutig (Bsp.: `while true do skip; x := 1;`).

Abstrakte Syntax liefert Bäume.

Mehrdeutigkeiten werden durch Klammerung beseitigt (Bsp.: `(while true do skip); x := 1;`).

Der Zustand ist eine Abbildung von Variablen auf ihre Werte:

$$\Sigma = \text{Var} \rightarrow \mathbb{Z} \quad \sigma \in \Sigma$$

Notation:

$$\sigma[x \mapsto n]$$

$\gg \sigma$ mit x abgebildet auf $n \ll$

$$\sigma[x \mapsto n](y) = \begin{cases} \sigma(y) & y \neq x \\ n & y = x \end{cases}$$

Beispiel: Konkretes σ : $\{x \mapsto 5, y \mapsto 7, \dots\}$

1.1 Semantik von Ausdrücken

$$\mathcal{A} : A\text{exp} \rightarrow (\Sigma \rightarrow \mathbb{Z})$$

$$\mathcal{B} : B\text{exp} \rightarrow (\Sigma \rightarrow T)$$

Definition mit Rekursion über die Syntax:

$$\begin{aligned} \mathcal{A}[[n]]\sigma &= n \\ \mathcal{A}[[x]]\sigma &= \sigma(x) \\ \mathcal{A}[[a_1 \text{ aop } a_2]]\sigma &= [[\text{aop}]](\mathcal{A}[a_1]\sigma, \mathcal{A}[a_2]\sigma) \\ &\quad [[\text{aop}]] : \mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{Z} \\ \mathcal{B}[[\text{true}]]\sigma &= \text{true} \\ \mathcal{B}[[a_1 \text{ rop } a_2]]\sigma &= [[\text{rop}]](\mathcal{A}[a_1]\sigma, \mathcal{A}[a_2]\sigma) \\ &\quad [[\text{rop}]] : \mathbb{Z} \times \mathbb{Z} \rightarrow T \\ \mathcal{B}[[b_1 \text{ bop } b_2]]\sigma &= [[\text{bop}]](\mathcal{B}[b_1]\sigma, \mathcal{B}[b_2]\sigma) \\ &\quad [[\text{bop}]] : T \times T \rightarrow T \end{aligned}$$

Beispiel

$$\begin{aligned} \mathcal{A}[[x + 5]]\{x \mapsto 7, \dots\} &= [[+]](\mathcal{A}[[x]]\{x \mapsto 7, \dots\}, \mathcal{A}[[5]]\{x \mapsto 7, \dots\}) \\ &= [[+]](7, 5) \\ &= 12 \end{aligned}$$

Fakt: \mathcal{A} und \mathcal{B} sind total.

Beweis: (zwei Möglichkeiten)

- offensichtlich primitiv rekursiv
- Induktion über die Struktur von $A\text{exp}$ und $B\text{exp}$.

Strukturelle Induktion: Zeige, dass eine Eigenschaft für einen zusammengesetzten Ausdruck gilt, vorausgesetzt, sie gilt für alle Teilausdrücke.

Kapitel 2

Operationale Semantik

Ziel: abstrakte Beschreibung von Berechnungen.

Methode: Inferenzregeln, die Zustandstransitionen beschreiben.

2.1 Big – Step Semantik (natural evaluation)

Transition: $\langle s, \sigma \rangle \rightarrow \sigma'$ mit:

- s ... Programm
- σ ... Anfangszustand
- σ' ... Endzustand

»Eigentlich« ist $\rightarrow \subseteq \text{Stm} \times \Sigma \times \Sigma$.

Regeln (Prolog)

$$\begin{array}{c} \langle \text{skip}, \sigma \rangle \rightarrow \sigma \\ \langle x := a, \sigma \rangle \rightarrow \sigma(x \mapsto \mathcal{A}[[a]]\sigma) \\ \frac{\langle s_1, \sigma_1 \rangle \rightarrow \sigma_2 \quad \langle s_2, \sigma_2 \rangle \rightarrow \sigma_3}{\langle s_1; s_2, \sigma_1 \rangle \rightarrow \sigma_3} \\ \frac{\mathcal{B}[[b]]\sigma = \text{true} \quad \langle s_1, \sigma \rangle \rightarrow \sigma'}{\langle \text{if } b \text{ then } s_1 \text{ else } s_2, \sigma \rangle \rightarrow \sigma'} \\ \frac{\mathcal{B}[[b]]\sigma = \text{false}}{\langle \text{while } b \text{ do } s, \sigma \rangle \rightarrow \sigma} \\ \frac{\mathcal{B}[[b]]\sigma = \text{true} \quad \langle s, \sigma \rangle \rightarrow \sigma' \quad \langle w, \sigma' \rangle \rightarrow \sigma''}{\langle w, \sigma \rangle \rightarrow \sigma''} \end{array}$$

mit $w = \text{while } b \text{ do } s$.

Bemerkung:

Format: $\frac{\text{Prämisse}}{\text{Konklusion}}$

Axiom: Regel ohne Prämisse

Regeln können zu Bäumen zusammengesetzt werden.

$$\begin{array}{l} w = \text{while } x \neq 1 \text{ do } x := x + 1 \\ \sigma = \{x \mapsto n\} \end{array}$$

Termination:

$$\frac{\mathcal{B}[[x \neq 1]]\sigma_1 = \text{false}}{\mathcal{B}[[x \neq 1]]\sigma_0 = \text{true}} \\ \langle x = x + 1, \sigma_0 \rangle \mapsto \sigma_1 \\ \frac{\langle w, \sigma_1 \rangle \mapsto \sigma_1}{\langle w, \sigma_2 \rangle \mapsto \sigma_1}$$

Nicht – Termination:

$$\langle w, \sigma_2 \rangle \mapsto ?$$

Fact: In einer Big – Step – Semantik drückt sich die Nicht – Termination von S , gestartet in σ dadurch aus, dass es kein σ' gibt mit $\langle s, \sigma \rangle \mapsto \sigma'$.

Definition: Programmäquivalenz bezüglich \mapsto :

$$S_1 \sim S_2 \Leftrightarrow \forall \sigma, \sigma' : \langle s_1, \sigma \rangle \mapsto \sigma' \\ \Leftrightarrow \langle s_2, \sigma \rangle \mapsto \sigma'$$

Lemma:

Sei $w = \text{while } b \text{ do } s$ und $w' = \text{if } b \text{ then } s; w \text{ else skip}$. Dann ist $w \sim w'$.

Beweis durch Transformation der Ableitungsbaume

F.U. nach $\mathcal{B}[b]\sigma$, dann Analyse der zuletzt benutzen Regel

•

$$\frac{\mathcal{B}[b]\sigma = \text{false}}{\langle w, \sigma \rangle \mapsto \sigma} \Leftrightarrow \frac{\mathcal{B}[b]\sigma = \text{false}}{\langle w', \sigma \rangle \mapsto \sigma} \frac{\langle \text{skip}, \sigma \rangle \mapsto \sigma}$$

• $\mathcal{B}[b]\sigma = \text{true}$

$$\frac{\frac{T_1}{\mathcal{B}[b]\sigma = \text{true}} \frac{\langle s, \sigma \rangle \mapsto \sigma''}{\langle w, \sigma \rangle \mapsto \sigma'}}{\langle w, \sigma \rangle \mapsto \sigma'} \Leftrightarrow \frac{\frac{T_1}{\langle s, \sigma \rangle \mapsto \sigma'} \frac{T_2}{\langle w, \sigma'' \rangle \mapsto \sigma'}}{\mathcal{B}[b]\sigma = \text{true}} \frac{\langle s; w, \sigma \rangle \mapsto \sigma'}{\langle w', \sigma \rangle \mapsto \sigma'}$$

Theorem:

$$\langle s, \sigma \rangle \mapsto \sigma_1 \wedge \langle s, \sigma \rangle \mapsto \sigma_2 \Rightarrow \sigma_1 = \sigma_2$$

d.h. die Semantik ist deterministisch.

Beweis: mit struktureller Induktion über s .

Nur 1 Fall: $s = \text{while } b \text{ do } s_0, \mathcal{B}[b]\sigma = \text{true}$

$$\frac{\langle s_0, \sigma \rangle \mapsto \sigma'_1 \quad \langle s, \sigma'_1 \rangle \mapsto \sigma_1}{\langle s, \sigma \rangle \mapsto \sigma_1} \wedge \frac{\langle s_0, \sigma \rangle \mapsto \sigma'_2 \quad \langle s, \sigma'_2 \rangle \mapsto \sigma_2}{\langle s, \sigma \rangle \mapsto \sigma_2}$$

IH: $\Rightarrow \sigma'_1 = \sigma'_2 \Rightarrow \sigma_1 = \sigma_2$

IH Unfug, IH stützt sich auf Annahme, die bewiesen werden soll.

Problem: Operationale Semantik von while ist nicht kompositional. While benutzt while, nicht nur Bedingung und Körper. \mathcal{A} und \mathcal{B} sind kompositional, d.h. primitiv rekursiv, d.h. die Bedeutung eines Konstrukts ist eine Funktion der Bedeutung der Teilkonstrukte.

2.2 Regelinduktion

Eine Menge von Regel der Form

$$\frac{a_1 \in X \dots a_n \in X}{a \in X}$$

(z.B. $(\text{skip}, \sigma, \sigma) \in \mapsto$) definiert induktiv die kleinste Menge X , die alle Regeln erfüllt.

Beispiel

$$0 \in \mathbb{N} \quad \frac{n \in \mathbb{N}}{n+1 \in \mathbb{N}} \quad \text{definiert } \mathbb{N}$$

Intuition: »kleinste Menge« \approx »kein Müll«

Beweise über induktiv definierte Mengen können mit Regelinduktion geführt werden.

Idee: $P(x)$ gilt für alle $x \in X$, falls P von jeder Regel erhalten wird:

$$P(a_1) \wedge \dots \wedge P(a_n) \Rightarrow P(a)$$

im Beispiel: für alle $n \in \mathbb{N}$: $P(n)$, falls

1. $P(0)$
2. $P(n) \Rightarrow P(n+1)$

Im Allgemeinen: für alle $x \in X$: $P(x)$, falls P von allen Regeln erhalten wird. $x \in X \Rightarrow P(x)$

Einfache Beispiele:

- Theorem: $n \in \mathbb{N} \Rightarrow n + n \in \mathbb{N}$
 Beweis mit Regelinduktion:
 R0: $P(0) = 0 + 0 \in \mathbb{N} = 0 \in \mathbb{N}$ ok (mit R0)
 R1: $P(n) \dots P(n+1)$

$$n + n \in \mathbb{N} \underset{\text{R1}}{\Rightarrow} n + n + 1 \in \mathbb{N} \underset{\text{R1}}{\Rightarrow} (n+1) + (n+1) \in \mathbb{N}$$

- Theorem: \mathbb{N} ist abgeschlossen unter der Addition: $m \in \mathbb{N} \wedge n \in \mathbb{N} \Rightarrow m + n \in \mathbb{N}$
 Beweis: $m \in \mathbb{N} \Rightarrow (n \in \mathbb{N} \Rightarrow n + m \in \mathbb{N}) = P(m)$ mit Regelinduktion
 R0: $n \in \mathbb{N} \Rightarrow 0 + n \in \mathbb{N}$ ok
 R1: $P(m) \Rightarrow P(m+1)$
 Gegeben: $n \in \mathbb{N} \Rightarrow m + n \in \mathbb{N}$
 zu zeigen: $n \in \mathbb{N} \Rightarrow (m+1) + n \in \mathbb{N}$

Transitiv reflexive Hlle Gegeben \rightarrow , definiere $\overset{*}{\rightarrow}$ induktiv:

$$\text{R0: } x \overset{*}{\rightarrow} x \quad \text{R1: } \frac{x \overset{*}{\rightarrow} y \quad y \rightarrow z}{x \overset{*}{\rightarrow} z}$$

Regelinduktion: $x \overset{*}{\rightarrow} y \Rightarrow Pxy$, falls

1. Pxx
2. $Pxy \wedge y \rightarrow z \Rightarrow x \overset{*}{\rightarrow} y$

Theorem: $x \rightarrow y \wedge y \overset{*}{\rightarrow} z \Rightarrow x \overset{*}{\rightarrow} z$

Beweis: mit Regelinduktion über

$$y \overset{*}{\rightarrow} z \Rightarrow (x \rightarrow y \Rightarrow x \overset{*}{\rightarrow} z)$$

R0: $P(x, x) = (x \rightarrow y \Rightarrow x \overset{*}{\rightarrow} y)$ ok (wegen Lemma)

R1: $P(x, y) \wedge y \rightarrow z \Rightarrow P(x, z)$

Annahme:

1. $a \rightarrow x \Rightarrow a \overset{*}{\rightarrow} y$ ($P(x, y)$)
2. $y \rightarrow z$

zu zeigen: $a \rightarrow x \Rightarrow a \xrightarrow{*} z (P(x, z))$

$$a \rightarrow x \Rightarrow a \xrightarrow{*} y \Rightarrow a \xrightarrow{*} z$$

Regelinduktion für Regeln mit Nebenbedingungen

$$\frac{a_1 \in X \dots a_n \quad \in XC_1 \dots C_m}{a \in X}$$

P bleibt unter dieser Regel erhalten genau dann, wenn

$$P(a_1) \wedge \dots \wedge P(a_n) \wedge C_1 \wedge \dots \wedge C_m \Rightarrow P(a)$$

Regelinduktion für $\langle s, \sigma \rangle \rightarrow \sigma' \Rightarrow P(s, \sigma, \sigma')$

1. $\forall \sigma : P(\text{skip}, \sigma, \sigma)$ (Regel: $\langle \text{skip}, \sigma \rangle \rightarrow \sigma$)
2. $P(x := a, \sigma, \sigma(x \mapsto \mathcal{A}[a]\sigma))$
3. $P(s_1, \sigma_1, \sigma_2) \wedge P(s_2, \sigma_2, \sigma_3) \Rightarrow P(s_1; s_2, \sigma_1, \sigma_3)$
4. $\mathcal{B}[b]\sigma \wedge P(s_1, \sigma, \sigma') \Rightarrow P(\text{if } b \text{ then } s_1 \text{ else } s_2, \sigma, \sigma')$
5. $\neg \mathcal{B}[b]\sigma \Rightarrow P(w, \sigma, \sigma)$
6. $\mathcal{B}[b]\sigma_1 \wedge P(s, \sigma_1, \sigma_2) \wedge P(w, \sigma_2, \sigma_3) \Rightarrow P(w, \sigma_1, \sigma_3)$

Theorem: while ist deterministisch:

$$\langle s, \sigma \rangle \rightarrow \sigma' \Rightarrow (\forall \sigma'' : \langle s, \sigma \rangle \rightarrow \sigma'' \Rightarrow \sigma' = \sigma'')$$

Beweis: mit Regelinduktion

7. $\mathcal{B}[b]\sigma_1$

$\text{IH}_s: \forall \sigma'' : \langle s, \sigma_1 \rangle \rightarrow \sigma'' \Rightarrow \sigma_2 = \sigma'$

$\text{IH}_w: \forall \sigma'' : \langle w, \sigma_2 \rangle \rightarrow \sigma'' \Rightarrow \sigma_2 = \sigma''$

zu zeigen: $\forall \sigma'' : \langle w, \sigma_1 \rangle \rightarrow \sigma'' \Rightarrow \sigma_3 = \sigma''$

Annahme: $\langle w, \sigma \rangle \rightarrow \sigma''$

\Rightarrow letzte Regel muss 7 gewesen sein

$\Rightarrow \exists \sigma'_1 : \langle s, \sigma_1 \rangle \rightarrow \sigma'_1 \wedge \langle w, \sigma'_1 \rangle \rightarrow \sigma''$

$\stackrel{\text{IH}_s}{\Rightarrow} \sigma'_1 = \sigma_2$, d.h. $\langle w, \sigma_2 \rangle \rightarrow \sigma''$

$\stackrel{\text{IH}_w}{\Rightarrow} \sigma'' = \sigma_3$

2.3 Small – Step – Semantik (structural operations)

Format: $\langle s, \sigma \rangle \rightarrow_1 \gamma$, wobei

- entweder $\gamma = \langle s', \sigma' \rangle$
noch keine Termination, »Restanweisung« s'
- oder $\gamma \in \Sigma$
Termination, Endzustand γ

Terminologie: $\langle s, \sigma \rangle$ BLOCKIERT, wenn es kein γ gibt mit $\langle s, \sigma \rangle \rightarrow_1 \gamma$.

Regeln

$$\begin{array}{c}
\langle \text{skip}, \sigma \rangle \rightarrow_1 \sigma \\
\langle x := a, \sigma \rangle \rightarrow_1 \sigma(x \mapsto \mathcal{A}[a]\sigma) \\
\frac{\langle s_1, \sigma \rangle \rightarrow_1 \langle s'_1, \sigma' \rangle}{\langle s_1; s_2, \sigma \rangle \rightarrow \langle s'_1; s_2, \sigma' \rangle} \\
\frac{\langle s_1, \sigma \rangle \rightarrow_1 \sigma'}{\langle s_1; s_2, \sigma \rangle \rightarrow \langle s_2, \sigma' \rangle} \\
\frac{\mathcal{B}[b]\sigma}{\langle \text{if } b \text{ then } s_1 \text{ else } s_2, \sigma \rangle \rightarrow_1 \langle s_1, \sigma \rangle} \\
\frac{\neg \mathcal{B}[b]\sigma}{\langle \text{if } b \text{ then } s_1 \text{ else } s_2, \sigma \rangle \rightarrow_1 \langle s_2, \sigma \rangle} \\
\langle w, \sigma \rangle \rightarrow_1 \langle \text{if } b \text{ then } s; w \text{ else skip}, \sigma \rangle
\end{array}$$

Eine REDUKTIONSSSEQUENZ für $\langle s, \sigma \rangle =: \gamma_0$ ist

- entweder eine endliche Sequenz $\gamma_0 \rightarrow_1 \gamma_1 \rightarrow_1 \dots \rightarrow_1 \gamma_n$ (Termination)
- oder eine unendliche Sequenz $\gamma_0 \rightarrow_1 \gamma_1 \rightarrow_1 \dots$ (Divergenz)

Jeder Schritt $\gamma_i \rightarrow \gamma_{i+1}$ muss mit den Regeln gerechtfertigt, d.h. hergeleitet werden.

Beispiel: $\sigma = \{x \mapsto 1, y \mapsto 2\}$

$$\begin{aligned}
\langle (z := x; x := y); y := z, \sigma \rangle &\rightarrow_1 \langle x := y; y := z, \sigma(z \mapsto 1) \rangle \\
&\rightarrow_1 \langle y := z, \sigma(z \mapsto 1, x \mapsto 2) \rangle \\
&\rightarrow_1 \{x \mapsto 2, y \mapsto 1, z \mapsto 1\}
\end{aligned}$$

Beispiel: $w = \text{while } x \neq 1 \text{ do } x := x + 1$ ($\sigma_n = \{x \mapsto n\}$)

$$\begin{aligned}
\langle w, \sigma_0 \rangle &\rightarrow_1 \langle \text{if } x \neq 1 \text{ then } x := x + 1; w \text{ else skip}, \sigma_0 \rangle \\
&\rightarrow_1 \langle x := x + 1; w, \sigma_0 \rangle \\
&\rightarrow_1 \langle w, \sigma_1 \rangle \\
&\rightarrow_1 \langle \text{if } x \neq 1 \text{ then } x := x + 1; w \text{ else skip}, \sigma_1 \rangle \\
&\rightarrow_1 \langle \text{skip}, \sigma_1 \rangle \\
&\rightarrow_1 \sigma_1 \\
\\
\langle w, \sigma_2 \rangle &\rightarrow_1 \langle \text{if } x \neq 1 \text{ then } x := x + 1; w \text{ else skip}, \sigma_2 \rangle \\
&\rightarrow_1 \langle x := x + 1; w, \sigma_2 \rangle \\
&\rightarrow_1 \langle w, \sigma_3 \rangle \\
&\rightarrow_1 \langle x := x + 1; w, \sigma_3 \rangle \\
&\rightarrow_1 \langle w, \sigma_4 \rangle \\
&\rightarrow_1 \dots
\end{aligned}$$

Lemma: Es gibt keine blockierenden $\langle s, \sigma \rangle$

Beweis: mit Induktion über s

- skip, $x := a$, while, if ok
- $s = s_1; s_2$
IH: s_1 blockiert nicht: $\langle s_1, \sigma \rangle \rightarrow \gamma$
 - $\gamma = \langle s'_1, \sigma' \rangle$: $\langle s_1; s_2, \sigma \rangle \rightarrow \langle s'_1; s_2, \sigma' \rangle$
 - $\gamma \in \Sigma$: $\langle s_1; s_2, \sigma \rangle \rightarrow \langle s_2, \sigma' \rangle$

Lemma:

$$\langle s_1; s_2, \sigma \rangle \rightarrow_1^n \sigma'' \Rightarrow \exists i, j, \sigma' : \langle s_1, \sigma \rangle \rightarrow_1^i \sigma' \wedge \langle s_2, \sigma' \rangle \rightarrow_1^j \sigma'' \wedge i + j = n$$

Beweis mit Induktion über n :

Basis $n = 0$ ist trivialerweise korrekt.

Schritt $n \rightarrow n + 1$: $\langle s_1; s_2, \sigma \rangle \rightarrow_1 \gamma \rightarrow_1^n \sigma''$

Fallunterscheidung nach der Regel, die im ersten Schritt benutzt wurde:

1.

$$\frac{\langle s_1, \sigma \rangle \rightarrow_1 \sigma'}{\langle s_1; s_2, \sigma \rangle \rightarrow_1 \langle s_2, \sigma' \rangle}$$

Dann gilt das Lemma mit $i = 1$ und $j = n$.

2.

$$\frac{\langle s_1, \sigma \rangle \rightarrow_1 \langle s'_1, \sigma_1 \rangle}{\langle s_1; s_2, \sigma \rangle \rightarrow_1 \langle s'_1; s_2, \sigma_1 \rangle}$$

Nach der Induktionshypothese existieren i, j, σ' : $\langle s'_1, \sigma_1 \rangle \rightarrow_1^i \sigma' \wedge \langle s_2, \sigma' \rangle \rightarrow_1^j \sigma'' \wedge i + j = n$.

Dann folgt die Behauptung für $n + 1$, denn $\langle s_1, \sigma \rangle \rightarrow_1^{i+1} \sigma' \wedge \langle s_2, \sigma' \rangle \rightarrow_1^j \sigma''$.

Theorem:

$$\langle s, \sigma \rangle \rightarrow \sigma' \Leftrightarrow \langle s, \sigma \rangle \rightarrow_1^* \sigma'$$

Beweis: $\Rightarrow \Leftarrow$: $\langle s, \sigma \rangle \rightarrow_1^n \sigma' \Rightarrow \langle s, \sigma \rangle \rightarrow \sigma'$

Induktion über n : Annahme: es gilt für alle $m < n$. Zeite. es gilt für n .

Annahme: $\langle s, \sigma \rangle \rightarrow_1^n \sigma'$. Dann gilt $n > 0$.

Fallunterscheidung über s :

- $s = \text{skip}$:

$$\sigma' = \sigma \Rightarrow \langle \text{skip}, \sigma \rangle \rightarrow \sigma'$$

- $s = s_1; s_2$:

$$\stackrel{\text{Lemma}}{\Rightarrow} \exists i, j, \hat{\sigma} : \langle s_1, \sigma \rangle \rightarrow_1^i \hat{\sigma} \wedge \langle s_2, \hat{\sigma} \rangle \rightarrow_1^j \sigma' \wedge i + j = n$$

$$\stackrel{\text{IH}}{\Rightarrow} \langle s_1, \sigma \rangle \rightarrow \hat{\sigma} \wedge \langle s_2, \hat{\sigma} \rangle \rightarrow \sigma'$$

$$\Rightarrow \langle s_1; s_2, \sigma \rangle \rightarrow \sigma'$$

- $s = \text{while } b \text{ do } s_0$:

$$\langle s, \sigma \rangle \rightarrow_1 \langle \text{if } b \text{ then } s_0; s \text{ else skip}, \sigma \rangle \rightarrow_1^{n-1} \sigma'$$

$$\Rightarrow \langle \text{if } b \text{ then } s_0; s \text{ else skip}, \sigma \rangle \rightarrow \sigma'$$

$$\Rightarrow \langle s, \sigma \rangle \rightarrow \sigma'$$

2.4 Erweiterungen von While

2.4.1 Nichtdeterminismus

Neue Anweisung: $s_1 \text{ or } s_2$.

Beispiel:

$$\langle x := 1 \text{ or } (x := 2; x := x + 4), \sigma \rangle \rightarrow \begin{cases} \sigma(x \mapsto 1) \\ \sigma(x \mapsto 6) \end{cases}$$

Big – Step:

$$\frac{\langle s_i, \sigma \rangle \rightarrow \sigma'}{\langle s_1 \text{ or } s_2, \sigma \rangle \rightarrow \sigma'} \quad i = 1, 2$$

Abkürzung:

$$\Omega := (\text{while true do skip})$$

$$\langle x := 1 \text{ or } \Omega, \sigma \rangle \rightarrow \sigma(x \mapsto 1)$$

Small – Step:

$$\langle s_1 \text{ or } s_2, \sigma \rangle \rightarrow_1 \begin{cases} \langle s_1, \sigma \rangle \\ \langle s_2, \sigma \rangle \end{cases}$$

Vergleich:

Big – Step – Semantik verdeckt Nichttermination. Small – Step – Semantik erhält Nichttermination.

2.4.2 Parallelismus mit gemeinsamen Variablen

Neues Konstrukt: $s_1 \parallel s_2$

Small – Step:

$$\frac{\langle s_1, \sigma \rangle \rightarrow_1 \langle s'_1, \sigma' \rangle}{\langle s_1 \parallel s_2, \sigma \rangle \rightarrow_1 \langle s'_1 \parallel s_2, \sigma' \rangle} \quad \frac{\langle s_2, \sigma \rangle \rightarrow_1 \langle s'_2, \sigma' \rangle}{\langle s_1 \parallel s_2, \sigma \rangle \rightarrow_1 \langle s_1 \parallel s'_2, \sigma' \rangle}$$

$$\frac{\langle s_1, \sigma \rangle \rightarrow_1 \sigma'}{\langle s_1 \parallel s_2, \sigma \rangle \rightarrow_1 \langle s_2, \sigma' \rangle} \quad \frac{\langle s_2, \sigma \rangle \rightarrow_1 \sigma'}{\langle s_1 \parallel s_2, \sigma \rangle \rightarrow_1 \langle s_1, \sigma' \rangle}$$

Beispiel:

$$\langle x := 1 \parallel (x := 2; x := x + 4), \sigma \rangle \rightarrow_1^* \sigma' \Rightarrow \sigma'(x) \in \{1, 5, 6\}$$

Big – Step:

$$\overline{\langle s_1 \parallel s_2, \sigma \rangle \rightarrow \sigma'} \quad \text{nicht möglich}$$

2.4.3 Lokale Variablen

Neues Konstrukt: $\{x := a; s\}$

x ist lokal in s .

Big – Step:

$$\frac{\langle s, \sigma(x \mapsto \mathcal{A}[a]\sigma) \rightarrow \sigma' \rangle}{\langle \{x := a; s\}, \sigma \rangle \rightarrow \sigma'(x \mapsto \sigma(x))}$$

Small – Step:

$$\langle \{x := a; s\}, \sigma \rangle \rightarrow_1 \langle x := a; s; x := \sigma(x), \sigma \rangle$$

$$\langle \{x := a; s\}, \sigma \rangle \rightarrow_1 \langle x' := a; s[x'/x], \sigma \rangle$$

(x' ist »neu«, Ersetzung von x durch x' in s)

2.5 Ein Compiler

Zielsprache: idealisierter Assembler

Instruktionen:

```

1  SET X a
2  JMP i           // relativer Sprung, i ganzzahlig
   JMPF b i       // relativer Sprung falls b = false

```

Semantik: Relation zwischen $\text{PC} \times \Sigma$

Notation: $P \vdash \langle i, \sigma \rangle \rightarrow \langle j, \sigma' \rangle$ mit P : Liste von Instruktionen und $i, j \in \mathbb{Z}$.

Regeln für $\vdash \langle \rangle \rightarrow \langle \rangle$:

Notation für Listen:

- P_n : n -tes Element von P (Listen starten mit 0)
- @: Konkatenation von Listen

$$\frac{P_n = \text{SET } Xa \quad 0 \leq n \leq |P|}{P \vdash \langle n, \sigma \rangle \rightarrow \langle n+1, \sigma(X \mapsto \mathcal{A}[a]\sigma) \rangle}$$

$$\frac{P_n = \text{JMP } i \quad 0 \leq n \leq |P|}{P \vdash \langle n, \sigma \rangle \rightarrow \langle n+i, \sigma \rangle}$$

$$\frac{P_n = \text{JMPF } b \ i \quad 0 \leq n \leq |P| \quad \mathcal{B}[b] = \text{true/false}}{P \vdash \langle n, \sigma \rangle \rightarrow \langle n+1/n+i, \sigma \rangle}$$

Notation:

$$P \vdash c \xrightarrow{n} c_n \Leftrightarrow \exists c_1, \dots, c_{n-1} : P \vdash c_i \rightarrow c_{i+1} \quad i = 0, \dots, n-1$$

Der Compiler

$$\begin{aligned} \text{comp} : \text{Stm} &\rightarrow \text{Instruktionsliste} \\ \text{comp skip} &= [] \\ \text{comp}(X := a) &= [\text{SET } Xa] \\ \text{comp}(s_1; s_2) &= \text{comp } s_1 @ \text{comp } s_2 \\ \text{comp}(\text{if } b \text{ then } s_1 \text{ else } s_2) &= \text{let } i_1 = \text{comp } s_1; i_2 = \text{comp } s_2 \\ &\quad \text{in}[\text{JMPF } b \ (|i_1| + 2)] @ i_1 @ [\text{JMP } (|i_2| + 1)] @ i_2 \\ \text{comp } w &= \text{let } i = \text{comp } s \\ &\quad \text{in}[\text{JMPF } b \ (|i| + 2)] @ i @ [\text{JMP } -(|i| + 1)] \end{aligned}$$

Compiler – Korrektheit

Lemma

1.

$$p \vdash \langle i, \sigma \rangle \xrightarrow{n} \langle j, \sigma' \rangle \Rightarrow p @ p' \vdash \langle i, \sigma \rangle \xrightarrow{*} \langle j, \sigma' \rangle$$

2.

$$p \vdash \langle i, \sigma \rangle \xrightarrow{n} \langle j, \sigma' \rangle \Rightarrow p' @ p \vdash \langle |p'| + i, \sigma \rangle \xrightarrow{*} \langle |p'| + j, \sigma' \rangle$$

Beweis: mit Induktion über n .

Theorem

$$\langle s, \sigma \rangle \rightarrow \sigma' \Rightarrow \text{comp } s \vdash \langle 0, \sigma \rangle \xrightarrow{*} \langle |\text{comp } s|, \sigma' \rangle$$

Beweis: mit Regelinduktion über $\langle s, \sigma \rangle \rightarrow \sigma'$

- Fall $\frac{\langle s_1, \sigma \rangle \rightarrow \sigma'' \quad \langle s_2, \sigma'' \rangle \rightarrow \sigma'}{\langle s_1; s_2, \sigma \rangle \rightarrow \sigma'}$

Abkürzung: $P_i = \text{comp } s_i$

IH₁: $P_1 \vdash \langle 0, \sigma \rangle \xrightarrow{*} \langle |P_1|, \sigma'' \rangle$

IH₂: $P_2 \vdash \langle 0, \sigma'' \rangle \xrightarrow{*} \langle |P_2|, \sigma' \rangle$

zu zeigen: $P_1 @ P_2 \vdash \langle 0, \sigma \rangle \xrightarrow{*} \langle |P_1| + |P_2|, \sigma' \rangle$

$$\begin{aligned} \text{IH}_1 &\Rightarrow P_1 @ P_2 \vdash \langle 0, \sigma \rangle \xrightarrow{*} \langle |P_1|, \sigma'' \rangle \\ &\Rightarrow P_1 @ P_2 \vdash \langle |P_1|, \sigma'' \rangle \xrightarrow{*} \langle |P_1| + |P_2|, \sigma' \rangle \end{aligned}$$

- Fall $\frac{\mathcal{B}[b]\sigma = \text{true} \quad \langle s, \sigma \rangle \rightarrow \sigma' \quad \langle w, \sigma'' \rangle \rightarrow \sigma'}{\langle w, \sigma \rangle \rightarrow \sigma'}$
 Abkürzungen: $P = \text{comp } s$, $W = \text{comp } w$
 $\text{IH}_s: P \vdash \langle 0, \sigma \rangle \xrightarrow{*} \langle |P|, \sigma'' \rangle$
 $\text{IH}_w: W \vdash \langle 0, \sigma'' \rangle \xrightarrow{*} \langle |W|, \sigma' \rangle$
 zu zeigen: $W \vdash \langle 0, \sigma \rangle \xrightarrow{*} \langle |W|, \sigma' \rangle$

$$\begin{aligned} \text{IH}_s &\Rightarrow W \vdash \langle 1, \sigma \rangle \xrightarrow{*} \langle |P| + 1, \sigma'' \rangle \quad (\text{Lemma}) \\ &\Rightarrow W \vdash \langle 0, \sigma \rangle \xrightarrow{*} \langle 0, \sigma'' \rangle \quad (\text{Semantik von JMP}) \\ &\stackrel{\text{IH}_w}{\Rightarrow} W \vdash \langle 0, \sigma \rangle \xrightarrow{*} \langle |W|, \sigma' \rangle \end{aligned}$$

Definition Eine Instruktionsliste P heißt ABGESCHLOSSEN genau dann, wenn alle Sprünge in P Ziele in $\{0, \dots, |P|\}$ haben.

Lemma $\text{comp } s$ ist abgeschlossen.

Lemma Sei c abgeschlossen.

$$\begin{aligned} P@c@P' \vdash \langle |P| + i, \sigma \rangle &\xrightarrow{n} \langle j, \sigma' \rangle \wedge 0 \leq i \leq |c| \wedge \neg(|P| \leq j \leq |P| + |c|) \\ \Rightarrow \exists \bar{\sigma}, n_1, n_2 : c \vdash \langle i, \sigma \rangle &\xrightarrow{n_1} \langle |c|, \bar{\sigma} \rangle \wedge P@c@P' \vdash \langle |P| + |c|, \bar{\sigma} \rangle \xrightarrow{n_2} \langle j, \sigma' \rangle \wedge n = n_1 + n_2 \end{aligned}$$

Beweis: mit Induktion über die Länge der Ausführungsfolge.

Lemma Sei c abgeschlossen.

$$\begin{aligned} P@c@P' \vdash \langle |P| + i, \sigma \rangle &\xrightarrow{n} \langle j, \sigma' \rangle \wedge 0 \leq i \leq |c| \wedge \neg(|P| \leq j \leq |P| + |c|) \\ \Rightarrow \exists \bar{\sigma}, n_1, n_2 : c \vdash \langle i, \sigma \rangle &\xrightarrow{n_1} \langle |c|, \bar{\sigma} \rangle \wedge P@c@P' \vdash \langle |P| + |c|, \bar{\sigma} \rangle \xrightarrow{n_2} \langle j, \sigma' \rangle \wedge n = n_1 + n_2 \end{aligned}$$

Theorem Sei $c = \text{comp } s$.

$$c \vdash \langle 0, \sigma \rangle \xrightarrow{*} \langle |c|, \sigma' \rangle \Rightarrow \langle s, \sigma \rangle \rightarrow \sigma'$$

Beweis mit Induktion über s :

Fall $s = \text{while } b \text{ do } s_0 \Rightarrow c = [\text{JMPF } b \dots]@ \text{comp } s_0@[\text{JMP } \dots]$.

Abkürzung: $c_0 = \text{comp } s_0$.

Zeige $c \vdash \langle 0, \sigma \rangle \xrightarrow{n} \langle |c|, \sigma' \rangle \Rightarrow \langle s, \sigma \rangle \rightarrow \sigma'$ mit Induktion über n .

Annahme: Es gilt für alle $m < n$. Zu zeigen: Es gilt für n .

Fallunterscheidung:

1. $\mathcal{B}[b]\sigma = \text{false} \Rightarrow \sigma' = \sigma \Rightarrow \langle s, \sigma \rangle \rightarrow \sigma'$

- 2.

$$\begin{aligned} \mathcal{B}[b]\sigma = \text{true} &\Rightarrow c \vdash \langle 0, \sigma \rangle \rightarrow \langle 1, \sigma \rangle \xrightarrow{n-1} \langle |c|, \sigma' \rangle \\ &\Rightarrow \exists \bar{\sigma}, n_1, n_2 : c_0 \vdash \langle 0, \sigma \rangle \xrightarrow{n_1} \langle c_0, \bar{\sigma} \rangle \wedge c \vdash \langle |c_0| + 1, \bar{\sigma} \rangle \xrightarrow{n_2} \langle |c|, \sigma' \rangle \\ &\quad \wedge n = n_1 + n_2 (\Rightarrow n_2 < n) \\ &\stackrel{\text{IH}_s}{\Rightarrow} \langle s_0, \sigma \rangle \rightarrow \bar{\sigma} \wedge c \vdash \langle 0, \bar{\sigma} \rangle \xrightarrow{n_2-1} \langle |c|, \sigma' \rangle \\ &\stackrel{\text{IH}_n}{\Rightarrow} \langle s, \bar{\sigma} \rangle \rightarrow \sigma' \\ &\Rightarrow \langle s, \sigma \rangle \rightarrow \sigma' \end{aligned}$$

Kapitel 3

Induktive Definition und Regelinduktion

Was sind Regeln?

$$\frac{a_1 \in X \dots a_n \in X}{a \in X}$$

Im Folgenden: $R \subseteq 2^A \times A$ Regelmengemenge, (H, a) ist Regel. Die Regelmengemenge darf unendlich sein, um schematische Regeln auszudrücken, z.B. ist $\frac{n \in X}{n+1 \in X}$ ein Regelschema.

Definition $B \subseteq A$ ist R -ABGESCHLOSSEN genau dann, wenn $\forall (H, a) \in R : H \subseteq B \Rightarrow a \in B$.

Definition Die durch R induktiv definierte Menge I_R ist die kleinste R -abgeschlossene Teilmenge von A .

Direkte Definition: $I_R := \bigcap \{B \subseteq A \mid B \text{ ist } R\text{-abgeschlossen}\}$

(Hinweis: Durchschnitt R -abgeschlossener Mengen ist wieder R -abgeschlossen.)

Intuition: »kleinste« ist kein Müll.

Nur das ist in I_R , was R erzwingt.

Beispiel $A = \mathbb{R}$

$$0 \in X, \frac{n \in X}{n+1 \in X}$$

kleinste R -abgeschlossene Teilmenge von \mathbb{R} : \mathbb{N} .

weitere R -abgeschlossene Teilmengen von \mathbb{R} : \mathbb{R}, \mathbb{Z} .

Mengentheoretisches Modell der Regeln:

$$R = \{(\{\}, 0)\} \cup \{(\{n\}, n+1) \mid n \in \mathbb{R}\}$$

3.1 Regelinduktion

Ziel:

$$\frac{\forall (\{h_1, \dots, h_n\}, a) \in R : P(h_1) \wedge \dots \wedge P(h_n) \Rightarrow P(a)}{\forall x \in I_R : P(x)}$$

Mengentheoretisch:

Lemma

$$I_R \subseteq P \Leftrightarrow I_R \cap P \text{ ist } R\text{-abgeschlossen}$$

denn

$$\begin{aligned} I_R \subseteq P &\Rightarrow I_R = I_R \cap P \Rightarrow I_R \cap P \text{ ist } R\text{-abgeschlossen} \\ I_R \cap P \text{ ist } R\text{-abgeschlossen} &\Rightarrow I_R \subseteq I_R \cap P \Rightarrow I_R \subseteq P \end{aligned}$$

Prädikatenlogisch:

$$(\forall x \in I_R : P(x)) \Leftrightarrow (\forall (H, a) \in R : H \subseteq I_R \wedge (\forall h \in H : P(h)) \Rightarrow a \in I_R \wedge P(a))$$

$a \in I_R$ folgt ohnehin aus $H \subseteq I_R$.

Nur \Leftarrow :

$$\frac{\forall (H, a) \in R : H \subseteq I_R \wedge (\forall h \in H : P(h)) \Rightarrow P(a)}{\forall x \in I_R : P(x)}$$

3.2 I_R als kleinster Fixpunkt

Definition R ist FINITÄR, falls H endlich ist für alle $(H, a) \in R$.

$$\begin{aligned} \hat{R} : 2^A &\rightarrow 2^A \\ \hat{R}(B) &= \{a : \exists H : (H, a) \in R \wedge H \subseteq B\} \end{aligned}$$

Ziel:

1. I_R ist kleinster Fixpunkt von \hat{R} .
- 2.

$$I_R = \bigcup_{i \in \mathbb{N}} \hat{R}^i(\{\})$$

Eigenschaften von \hat{R} :

1. B ist R -abgeschlossen $\Leftrightarrow \hat{R}(B) \subseteq B$
2. \hat{R} ist monoton ($B_1 \subseteq B_2 \Rightarrow \hat{B}_1 \subseteq \hat{B}_2$).

\hat{R} generiert I_R :

$$\begin{aligned} A_0 &:= \hat{R}^0(\{\}) = \{\} \\ A_1 &:= \hat{R}^1(\{\}) = \hat{R}(\{\}) \\ &\vdots \\ A_{n+1} &:= \hat{R}^{n+1}(\{\}) = \hat{R}(\hat{R}^n(\{\})) \\ &\vdots \\ A_\omega &:= \bigcup_{i \in \mathbb{N}} A_i \end{aligned}$$

Beispiel Regeln:

$$\begin{aligned} 0 \in E \quad \frac{n \in E}{n+2 \in E} \quad A = \mathbb{N} \\ \hat{R}(M) = \{m+2 \mid m \in M\} \cup \{0\} \end{aligned}$$

$$\begin{aligned}
\hat{R}^0(\{\}) &= \{\} \\
\hat{R}^1(\{\}) &= \{0\} \\
\hat{R}^2(\{\}) &= \{0, 2\} \\
\hat{R}^3(\{\}) &= \{0, 2, 4\} \\
&\vdots \\
A_\omega &= \{2n \mid n \in \mathbb{N}\}
\end{aligned}$$

Fakt Die A_i bilden eine ω -Kette $A_0 \subseteq A_1 \subseteq A_2 \subseteq \dots$
 Beweis mti Induktion:

$$\begin{aligned}
A_0 &= \{\} \subseteq A_0 \\
A_{i+1} &= \hat{R}(A_i) \subseteq \hat{R}(A_{i+1}) = A_{i+2}
\end{aligned}$$

Theorem Ist R finitär, dann gilt

- i. A_ω ist R -abgeschlossen ($\hat{R}(A_\omega) \subseteq A_\omega$)
- ii. $\hat{R}(A_\omega) = A_\omega$
- iii. $A_\omega = I_R$

Beweis:

- i. Sei $(\{h_1, \dots, h_n\}, a) \in R$ mit $H = \{h_1, \dots, h_n\} \subseteq A_\omega$.
 Zu zeigen: $a \in A_\omega$.

$$\begin{aligned}
&\forall 1 \leq i \leq n \exists k_i : h_i \in A_{k_i} \\
k &:= \max\{k_1, \dots, k_n\} \Rightarrow \forall 1 \leq i \leq n : h_i \in A_k \\
&\Rightarrow a \in A_{k+1} \subseteq A_\omega
\end{aligned}$$

- ii. Sei $a \in A_\omega$, d.h. $\exists i > 0 : a \in A_i = \hat{R}(A_{i-1}) \subseteq \hat{R}(A_\omega)$
- iii. Wir zeigen: A_ω ist die kleinste R -abgeschlossene Menge.
 Annahme: $B \subseteq A_\omega$, B ist R -abgeschlossen, d.h. $\hat{R}(B) \subseteq B$.
 Wir zeigen: $A_i \subseteq B$ mit Induktion über i :

$$A_{i+1} = \hat{R}(A_i) \subseteq \hat{R}(B) \subseteq B$$

Damit gilt auch $A_\omega \subseteq B$.

Korrolar Sei R finitär.

1. $I_R = \bigcup_{i \in \mathbb{N}} \hat{R}^i(\{\})$
2. I_R ist kleinster Fixpunkt von \hat{R} .

Beweis von 2.: Jeder Fixpunkt ist R -abgeschlossen:

$$\hat{R}(B) = B \Rightarrow \hat{R}(B) \subseteq B \Rightarrow I_R \subseteq B$$

Beispiel

$$0 \in X \quad \frac{n \in X}{n+1 \in X} \quad \frac{0 \in X, 1 \in X, \dots}{-1 \in X}$$

$$\begin{aligned}
I_R &= \mathbb{N} \cup \{-1\} \\
A_i &= \{0, \dots, i\} \quad A_\omega = \mathbb{N}
\end{aligned}$$